Accelerating Network Traffic Analytics Using Query-Driven Visualization

E. Wes Bethel[*]
Computational Research Division

Scott Campbell
National Energy Research Scientific Computing Center Division

Eli Dart
Energy Sciences Network

Kurt Stockinger
Computational Research Division

Kesheng Wu
Computational Research Division

Lawrence Berkeley National Laboratory
University of California
Berkeley, CA 94720

Abstract

Realizing operational analytics solutions where large and complex data must be analyzed in a time-critical fashion entails integrating many different types of technology. Considering the extreme scale of contemporary datasets, one significant challenge is to reduce the duty cycle in the analytics discourse process. This paper focuses on an interdisciplinary combination of scientific data management and visualization/analysis technologies targeted at reducing the duty cycle in hypothesis testing and knowledge discovery. We present an application of such a combination in the problem domain of network traffic data analysis. Our performance experiment results show that the combination can dramatically decrease the analytics duty cycle for this particular application. The combination is effectively applied to the analysis of network traffic data to detect distributed scans, which is a difficult-to-detect form of cyberattack. We reduce the analytics duty cycle by leveraging high-performance data management technology for indexing and querying data: fast query technology accelerates the mechanics of hypothesis testing, and straightforward visualization of readily available data distributions accelerates query formulation and knowledge discovery. Our approach is sufficiently general to be applied to a diverse set of data understanding problems as well as used in conjunction with a diverse set of analysis and visualization tools.

CR Categories: H.2.8.h [Interactive data exploration and discovery];I.6.9.d [Multivariate visualization]; K.6.M.b [Security]; J.8.o [Traffic Analysis]

Keywords: query-driven visualization, network security, data mining, visual analytics

1 Introduction

Visual Analytics is defined in [27] as "the science of analytical reasoning facilitated by active visual interfaces." It is motivated by the need to gain understanding of features, trends and anomalies present in large and complex data collections. While a thorough discussion of the immense scope of all possible technical challenge areas and motivations is well beyond the scope of this paper, interested readers are directed to [27], which is a broad survey of the current state of research and development challenges in the field. From that broad set of challenges, one in particular is the focus of this paper: how to quickly find "interesting data" in large, multidimensional collections of information. We explore this topic within the context of a cybersecurity application, namely network traffic analysis.

Network traffic datasets consist of records containing a number of variables that summarize a particular network connection, or "conversation" between two hosts on a network. These data – known as connection records – are generated by routers, traffic analyzers or security systems, and contain such information as source and destination IP address of the conversing hosts, the source and destination ports, duration of connection, number of bytes exchanged and so forth. There is no information about the traffic content, only information about the two hosts participating in the connection. With the explosive growth of the Internet, there is a corresponding rise in the amount of information collected about network connections as well as an increase in the number of anomalous events. Such events may be indicative of a misconfigured host or network, inappropriate use of resources, an attack on a computer system or network, a compromised host, or any one of a number of other items of interest. Collecting, managing and understanding the growing amount of network connection data in a timely fashion all present substantial challenges for network operations personnel.

A broad view of the network traffic analysis problem would necessarily include data collection, data storage and management, automatic feature detection, event characterization, analytical discourse to understand features and discover their relationships along with timely response to a particular incident. The work we present here explores a of subset the complete network traffic analysis problem. Namely, we focus on a multidisciplinary approach to feature mining and hypothesis testing by combining scientific data management tools for indexing and querying with simple visualization tools. The overall objective of our work is to reduce the duty cycle in hypothesis testing and feature mining. This paper describes how we achieve that objective within the context of a network traffic analysis case study. The results are particularly relevant given the explosive growth in network traffic and network traffic data. To be effective, a complete visual analytics solution will need to address problems of scale, to make effective use of high performance computing resources, and to quickly provide answers to analysts. Arguably, effective data management is the cornerstone for all such applications.

The main contributions of this paper are: (1) an integrated, cross-disciplinary approach combining state-of-the-art data management, well-understood visualization techniques and a straightforward interface results in a new data analytics capability that is highly effective and efficient; (2) application of this approach to a "hero-sized" network traffic data analysis problem.

The rest of this paper is structured as follows. Section 2 presents a survey of previous work in fields related to the topic of this paper. Since our work realizes new capability by integrating and applying ideas from several different fields in a multi- and inter-disciplinary fashion, we discuss prior work from several different but related fields. Section 3 presents an experiment profiling the performance of different technologies for performing queries on a realistic-sized collection of network traffic data. The main point of this experiment is to highlight the vastly different performance capabilities of

[*]e-mail: ewbethel@lbl.gov, scampbell@lbl.gov, dart@es.net, kstockinger@lbl.gov, kwu@lbl.gov

different technologies. Section 4 contains a network traffic analysis case study where the confluence of scientific data management and visualization comprise a visual analytics implementation. We capitalize upon fast queries vis a vis bitmap indexing to reduce the duty cycle in knowledge discovery. A cornerstone of our visual analytics approach is to use unique the characteristics of a bitmap indexing implementation, namely the ability to quickly generate data distribution graphs (histograms), and straightforward and easy-to-understand visualization to implement the visual analytics portion component of knowledge discovery and data mining. The network traffic analysis case study was designed by network analysts specifically using the new concepts resulting from the intersection of visualization and data management.

2 Background and Related Work

In recent years, there has been a tremendous increase in research and development projects in the areas of network traffic analysis and visualization. Related to the material we present here is previous work in the fields of scientific data management, particularly techniques for efficient indexing and querying, as well as visualization systems that focus on limiting visual processing only to data deemed to be "of interest" to the viewer.

2.1 Network Traffic Analysis

There exists an increasing need for visual analytics tools in the fields of network analysis and forensics. Raw traffic logs have long been too large and complex for a human to digest and understand. In particular, there is a need for tools that provide insight into patterns in data sets that are large in volume, time or both. Intrusion Detection Systems (IDS) are typically good at analyzing events that are closely correlated in time, and where analysis can rapidly yield actionable results. However, over large time scales and/or very large data sets, the design decisions that make IDS software appropriate for rapid response limit applicability to problems of scale. Typical failure modes include unbounded memory consumption or computational overload that inhibits a rapid response. Moreover, connection analysis lends itself to visual analytics, since features and patterns that are easily visible to a skilled analyst are often difficult to quantify precisely or to detect programmatically. Therefore, visual analytics tools are best thought of as a complement to IDS software, and part of a broad technology portfolio in the network analyst's toolbox.

Traditional high volume network traffic analysis usually begins after the IDS provides an alert, such as when one or more IP addresses are associated with a possible attack. If the analyst will be examining short-term connection data – less than 24 hours worth of data – then the connection logs are dumped straight to local disk. These logs can usually be queried in 10 seconds or less. For analysis that spans longer periods of time, a dedicated system is typically available that can process connection logs at the rate of 5-7 minutes per month for one or more IP addresses. The tool typically used for searching text-format connection records is grep. Once the subset of interesting connection data have been extracted from the larger set of logs, the analyst performs more specialized processing pursuant to the particular line of inquiry. Initial queries – search terms – based on anything other than a single column are rarely performed. Post-processing of search results is performed using tools such as perl, shell scripts and gnuplot. The time required for these steps is proportional to the amount of script development and analysis required for the particular incident.

Historically, the problem with large-scale analysis tools for network traffic has been that the duty cycle for testing a given hypothesis is measured in hours for non-trivial data sets. Therefore, a reduction in turnaround time from hours to seconds represents an unprecedented new capability. The new capability migrates large-scale network connection analysis from the realm of overnight batch jobs to that of interactive tools.

Visualization techniques permit an analyst to look at thousands or hundreds of thousands of connections simultaneously. Simple data filtering is both valuable and necessary to deal with the huge volume of data and the large dynamic range exhibited by many of the attributes. Applying multi-dimensional transformations, deriving statistical quantities and applying clustering techniques provides new and often more relevant quantities to visualize, as well as more relevant keys for indexing and querying.

A network connection can be thought of as a set of packets passing between two hosts within a given time interval that have common characteristics. An example of a network connection is a single communication session or an interaction between two hosts on the Internet. Several standard tools exist for capturing network connection data. Tcpdump is one of the most commonly used; it is a pcap-based application that can run on most operating systems and logs network traffic based on a filtering expression [7, 14]. For larger environments, routers and switches can provide connection data in specialized formats such as NetFlow [26] or SFlow [19]. Special purpose systems and software for analyzing NetFlow and SFlow records have been implemented by network equipment vendors, but these tools are typically optimized for aggregate network usage and trend analysis. Some network connection data collection and reporting systems generate a separate record for each direction of bidirectional connection [18] Other systems generate a single full-duplex connection record that also contains byte and packet counts for the reverse direction [28]. Individual network services like HTTP are application-level services built atop transport-level protocols (TCP, UDP).

For this discussion we describe connection dynamics in terms of TCP session characteristics. A network connection record generally contains at least the following information: 1. Source IP address, 2. Destination IP address, 3. Source Port, 4. Destination Port, 5. Byte and packet count sent by source, 6. Byte and packet count sent by destination, 7. Start and End time in milliseconds. 8. TCP state.

A typical day's worth of traffic at an "average" government research laboratory might involve tens of millions of such connections comprising multiple gigabytes worth of connection records. A year's worth of such data currently requires on the order of tens of terabytes or more of storage. According to Burrescia [3], traffic volume over ESnet, a production network servicing the U. S. Department of Energy's research laboratories, has increased by an order of magnitude every 46 months since 1990. This trend is expected to continue into the foreseeable future.

2.2 Network Traffic Visualization

There has been a good deal of work in recent years in the area of interactive network traffic visualization. A thorough survey of such work is outside the scope of this paper since we are focusing on coupling data management technology with network traffic visualization and analysis tools. See [13] for a good survey of previous work in this area. Generally speaking, previous work has focused on filtering and visual presentation of different types of network traffic data, including connection data, routing information, IDS alerts and so forth.

Visualization applications aimed at providing overall situational awareness by visualization network connection data are described in [12, 15]. These applications map network connection variables to axes, then present activity, or lack thereof, at the appropriate grid location. The basic idea is to facilitate rapid visual discovery of incidents or other features. [15] uses interactive filter manipulation to reduce the amount of data being displayed. In this way, a user can focus visual inspection only on data that matches a set of filtering criteria. In both these examples, the problem size consists of datasets comprised of a few thousand unique network connections.

[11] describes an Intrusion Detection Toolkit that supports a number of different techniques for viewing alert or packet data. This system implements data reduction through filtering. This work

is silent on the subject of filtering, or query methodology and performance, and shows examples of what appear to be small datasets.

The VisAlert system [13] presents a visualization paradigm for correlating network alerts generated by multiple sensors deployed across a network. The paradigm is based on the observation that every network alert must possess three fundamental attributes – what, when and where – that in turn provide a consistent basis for correlation. VisAlert uses a unique and flexible two-dimensional display metaphor that is effective in helping users to switch between context/focus modes of inspection. Their results use a dataset consisting of 12-hours' worth of network traffic.

While these previous works in network traffic visualization have produced novel and useful presentation and interaction techniques, they were tested using only small amounts of network data. In contrast, our work focuses on techniques suitable for use with large collections of network connection data. Our "filtering" is provided by highly optimized data management technology designed for rapid data mining of terascale datasets. Our visualization emphasizes straightforward and easy-to-understand techniques for rapid inspection of data distributions for the purposes of formulating queries. These two technologies are manipulated via an easy-to-use visual analytics interface. We believe such an approach is required to gain traction on analysis and visualization of realistic-sized volumes of network traffic and thus offers a completely new capability to the field of network traffic analysis and visualization.

2.3 Query-Driven Visualization

The term "Query-Driven Visualization" refers to the process of limiting visualization processing and subsequent visual interpretation to data that is deemed to be "interesting." Several factors contribute to the overall motivation for the query-driven visualization approach. As data grows larger and more complex, simply building larger and more scalable visualization systems produces a greater amount of output, which in turn increases the cognitive load on the viewer. In some cases, increasing the amount of visible output may actually hinder understanding as depth complexity increases, important features are "hidden," and so forth. Similarly, with increasing data size and complexity, finding and displaying relevant data becomes increasingly important to foster scientific understanding and insight. The query-driven visualization approach also offers a new foothold for gaining traction on the challenges of temporal and multidimensional visualization and analysis since processing is focused on "sets" of data that satisfy the conditions for a given line inquiry. An example here might be "what is the average electrical charge of particles in an accelerator model that that spin away from the main beamline and strike the accelerator wall over the course of the entire simulation?"

The VisDB system combines a guided query-formulation facility with relevance-based visualization [9]. Each data item in a dataset is ranked in terms of its relevance to a query, and the top quartile of results is then input to a visualization and rendering pipeline. Data is presented in a way to cluster more relevant items closer together, and less relevant items further apart. It is especially well-suited to display the results of "fuzzy queries" in that inexact matches are ranked and visually displayed in a way to convey relevance.

The TimeFinder system described in [6] supports interactive exploration of time-varying data sets by providing the ability to quickly construct queries, modify parameters, and visualize query results. A query is formed by manually "drawing" a rectangular box on a 2D plot where the x-axis represents time and the y-axis represents the data range. Each such rectangular box is called a "timebox" and comprises a range query. A user forms a multidimensional range query through the union of several timeboxes. The query results are presented in a fashion that implements a form of data mining – more detailed information about the items satisfying the query are presented in a separate window. TimeFinder reads all data into memory and is therefore able to operate on only modest-sized datasets.

Recently, the idea of coupling a visualization pipeline with a high performance index and query system was described [24]. That work shows that the computational complexity of visualization processing can be constrained to the number of items returned by a query. As such, that approach is the most suitable for use in query-driven visualization and analysis of very large multidimensional datasets.

We are extending the work of [24] in this paper by: applying indexing and querying techniques to network connection data; providing a performance comparison between a limited set of data mining technologies; capitalizing upon statistical information in the data management infrastructure to create easy-to-comprehend visual displays; combining all technologies into an integrated framework for rapid knowledge discovery; demonstrating the combination's applicability within the context of a "hero-sized" network connection data analysis problem.

2.4 Indexing and Querying

One approach for examining a large amount of network connection data is to focus attention on a relatively small number of "interesting" connections. The naive approach of checking every connection to see if it satisfies the definition of "interesting" may take too long or be impractical. The basic strategy to accelerate the selection process is to use an indexing structure [10]. Most well known indexing structures are designed for data that are frequently updated, like bank transactions. In banking applications when some arbitrary record is modified, the index structure must be similarly updated. In such applications, the query/update indexing structure functions must both both be very efficient. For many tree-based indexing structures, the time required for updating a record is nearly the same as the time required for locating a record. Network connection data is different from these types of transactional databases in that the existing records are never modified. The only change to the data is the addition of new records. It is possible to gain query efficiency by sacrificing update efficiency, and bitmap indexing technology is an example of technology with such a performance tradeoff.

Bitmap indexing has been implemented in commercial database systems and it is well accepted that it is efficient for low cardinality attributes where there are few distinct values [16]. Recently, it was shown that such efficiency can be extended to high cardinality attributes with Word Aligned Hybrid coding (WAH) [30]. FastBit [21] is a research code that implements a number of different forms of bitmap index compression, including WAH.

In a basic bitmap index, one bitmap is allocated for each distinct value of the indexed attribute, where each bitmap has as many bits as the number of records in the indexed dataset. The size of the index grows linearly with the attribute cardinality. There are a number of strategies for reducing the size of a bitmap index, including binning [22, 23], multi-component encoding [4], and compression [8, 29]. WAH compression was proven to keep the index sizes compact as well as to significantly reduce the query processing time compared to other indexing schemes [30].

In this paper, we compare the performance of a WAH compressed bitmap index with one called a projection index [17]. The projection index extracts the attribute values and stores them separately so that when answering a query, only the attributes involved in the query are read into memory. This approach is known to work well for range queries, which are commonly used for analysis of large datasets.

While exploring network connection data, an analyst might use a query of the form "StartTime > 20050501 AND 10.102.0.0 <= SourceIP <= 10.105.255.255." In this example, each term like "StartTime > 20050501" is called a range condition. In a typical exploration, the analyst may specify a number of different range conditions on different attributes. Such queries are typically referred to as "ad-hoc" since they do not follow a predefined pattern. With ad-hoc queries, the bitmap index has a unique advantage over tree-based indexing structures because the answers to individual range condition can be efficiently combined. Most tree-based indexing methods suffer from the "curse of dimensionality." As the number of attributes in a dataset increases, tree-based indices become progressively less competitive against the projection index. Typical multi-dimensional indexing tree-based – kd-trees, for example [1] – usually index all variables or dimensions of a dataset. When answering an ad-hoc range query involving only a few variables, tree-based multi-dimensional indices are much less efficient than the projection and bitmap indices, which do not suffer from the "curse of dimensionality."

To answer multidimensional range queries, we first use the bitmap indices to resolve each individual range condition and then combine the partial solutions with bitwise logical operations. The time required to resolve each range condition is proportional to the size of the bitmaps involved. Moreover, the overall query processing time grows linearly with the number of range conditions specified. The time required by the projection indices also scales linearly with the number of range conditions, however, the time required to resolve each individual range condition using a projection index is typically much longer than that of a bitmap index as we will empirically show later in Section 3.

3 Query-Driven Network Traffic Analysis Performance Study

Since most visual analytics analytics applications perform some type of data mining, increasing data mining performance is central to reducing the duty cycle in the analysis process. In this section we measure the performance of three different technologies for data mining.

We begin with three background sections. The first describes the network traffic data we use in the performance measurements as well as in Section 4's case study. The second identifies the experiment's computing environment. The third focuses on each of the three competing data mining implementations. The performance study consists of two sets of tests – serial and parallel – for each of the three competing technologies.

3.1 Network Traffic Data

The network traffic data in our performance experiment and network traffic analysis case study consists of 42 weeks' worth of network connection data collected from a Bro system running at NERSC consisting of about 2.5 billion records having a total size of about 281GB. Each record contains 25 attributes, including the source IP address, the destination IP address, source port, destination port, start time, duration, number of bytes sent along with additional network connection information. To increase query efficiency, we have split the IP addresses into four octets A, B, C and D. For instance, IPS_A refers to the class A octet of the source IP address. Data is stored in a one-week-per-file distribution – not necessarily the most efficient for parallel computing, but convenient for the analysts managing the data and interpreting the query results.

In addition to the raw data, the bitmap indices themselves require a total of about 78.6 GB of space. We created a standalone utility that creates the indices: it reads all network connection data and uses FastBit to generate compressed bitmap indices. It is worth noting that the 78.6 GB of space required for the indices is only about one-third the size of the raw data. This figure compares favorably to tree-based indices (e.g., B-trees) typically found in transactional database systems where the tree-based indices are typically three times the size of the raw data.

3.2 Computing Environment

We use a single computing platform for both the performance experiments and the network traffic analysis case study. The platform is a 32-processor SGI Altix with 192GB of RAM and 40TB of fiberchannel RAID capable of delivering 500GB/s in sustained I/O performance. We chose this platform due to its combination of vast amounts of memory and its high-performance I/O to secondary storage. Such platforms are well-suited to data intensive analysis and visualization tasks.

3.3 Query Implementation

For the purposes of measuring and comparing query response time, we use three different approaches. The first is representative of the type of technology typically used in production networks for traffic and security analysis. The second has found use in analysis of large collections of high-energy physics data and is based upon projection indices. The third is FastBit, which uses bitmap indexing, which is the form of indexing in use by all major commercial database systems.

The first approach is a series of scripts and shell-based tools (awk, grep, etc.) that parse and search through an ASCII version of the connection records. At first glance to one outside the network security business, this approach may seem to be naive. In fact, this approach is widely used in network traffic analysis to overcome limitations caused by proprietary data formats and the frequent need to perform ad-hoc analysis. Shell scripts and tools are easy to change, readily shareable, highly transportable across platforms and, of course, can be quite slow compared to compiled and optimized applications. They are relatively easy to write, which makes them ideal for fast "one-off" analysis tasks. That very strength becomes a liability when viewed from a software engineering perspective – shell-based are don't exhibit the type of modularity that promotes reuse and are not especially suitable for use on parallel platforms. Shell-based tools have $O(n)$ complexity – all data records must be examined in the search for those that meet a given set of criteria. Historically, network analysts typically work with relatively small collections of data – hours' or days' worth of traffic. For those small data sizes, tools based on shell scripts execute with a duty cycle on the order of 10s or 100s of seconds. Since we are tackling a much larger problem in this paper – 42 weeks' worth of network traffic – there is value in showing how existing approaches are not effective in reducing the analytics duty cycle.

For the second and third types of queries, our implementation is is based on ROOT [2], which is an object-oriented data analysis system originally developed for scientific analysis and data management of terascale volumes of high-energy physics data. ROOT is widely recognized as the "gold standard" for data mining and analysis in the high energy physics community where experiments routinely generate tens to hundreds of terabytes of data per year. Data analysis in that space typically consists of searching for a particle detector events that satisfy a very narrow range of criteria.

The ROOT system has a comprehensive set of analysis capabilities and basic visualization features and supports parallel operations – including parallel data mining and analysis. While its native multidimensional query engine is built using "projection indices," we extended ROOT so that it can answer queries using either its native projection indices or FastBit's compressed bitmap indices. To support this type of dual-mode use, we created two separate versions of the network traffic data – one organized specifically into ROOT's projection index format, and one that uses FastBit's native data storage format.

3.3.1 Serial Performance Comparison

To establish a performance baseline, we measure the time required to answer the following three-dimensional network traffic analysis query (i.e., a query comprised of three variables): "select IPS_B, IPS_C, IPS_D where $IBS_B < 100$ AND $IPS_C < 100$ AND $IPS_D = 128$." This query locates those network connections originating from a given range of IP addresses. We performing this query on 42 weeks' worth of network data.

The time required to answer this query using "typical network traffic analysis software" is approximately 51,000 seconds. [1] For the projection index test, we use ROOT with only its projection indices, i.e., without FastBit. In that configuration, the time required to answer the query is 1269 seconds. By switching to the FastBit's bitmap indices in ROOT, the time required to answer the query is 419 seconds.

By using FastBit's bitmap indices in our ROOT application, we realize a factor of three performance gain simply by migrating from projection to bitmap indices. While these results were run using network connection data as the source, the relative performance gain is consistent with previous work comparing multidimensional query performance on large collections of high energy physics data [25].

3.3.2 Parallel Performance Test

Like many other endeavors in high performance computing, data intensive operations stand to benefit from parallelism. We implemented a battery of tests to measure the performance of the three competing technologies when run in a parallel configuration. We ran tests using 1, 2, 6, 13 and 21 processors – these levels of parallelism were chosen so that each processor will be responsible for an equal number of weeks' worth of network traffic data. Since there is substantial variation in the amount of connection data from week to week, a by-week domain decomposition does not ensure even load balance in terms of computation or I/O. For these tests, we used a variant of a query that appears in Section 4: find all records where (DP==5554) and (STATE==1) and (IPS_A==220) and (IPS_B==184) and (IPS_C==26). The origin of this particular query and its importance appears in Section 4.

The results are summarized in Table 1. The first column shows the number of processing elements (PEs) used for our performance study. The second column shows the query response time for evaluating the 5-dimensional query with the shell-based approach. Columns three and four show the performance results of ROOT/Projection Index and ROOT/FastBit. As expected, the shell-based approach, which is most commonly used for network traffic analysis, is by far the slowest. ROOT/FastBit, on the other hand, shows excellent performance. With 21 processors, the time to analyze a 5-dimensional query over 42 weeks of network traffic data (281 GB) only takes about 2 seconds. Neither of the three approaches has optimal scalability due to data load imbalance. FastBit is especially well suited for these kinds of queries since only a small fraction of the bitmap index needs to be read to satisfy the query. The other two approaches must to scan the whole 281 GB of data. FastBit's computational complexity is at worst $O(h)$ where h is the number of items returned by the query [30], while the other two approaches have $O(n)$ complexity where n is the number of data records being searched.

4 Network Traffic Analysis Case Study

In this section, we present a case study illustrating how the combination of fast search/query operations combined with interactive analytics and visualization gives rise to a highly practical and efficient methodology for network traffic analysis. For this case study, we use the same 42 weeks' worth of network data and computing platform as in Section 3. The case study shows how an iterative process of visual inspection and data mining lead to the discovery of a distributed scan.

A distributed scan is a specialized form of a "port scanning" attack in which multiple distributed hosts systematically probe for vulnerabilities on a set of hosts. A specialized form of distributed scanning involves attacking hosts that have themselves been compromised and conscripted for use in a distributed scan attack by a third party. The third party acts as a "central authority" for managing the attack. This form of distributed scan is known as a "bot-net" attack.

[1] Our "shell scripts" consist of a mixture of "awk," "grep," run in a Bourne shell script. A substantial fraction of the 51,000 seconds we report is due to "awk" processing logic. Whether that processing time would be substantially reduced by using "perl" is an open question, but outside the scope of this paper.

PEs	Shell-based	ROOT/Projection Index	ROOT/FastBit
1	156381.14	1357.07	5.36
2	71835.32	600.05	3.72
6	21952.12	214.14	2.66
13	9389.96	113.88	2.58
21	2237.53	98.95	2.05

Table 1: Parallel performance evaluation of a 5-dimensional network traffic analysis query: Find all records where (DP==5554) and (STATE==1) and (IPS_A==220) and (IPS_B==184) and (IPS_C==26). The time is reported in seconds. ROOT-FastBit significantly outperforms the two competing approaches. Timings for the shell-based scripts are estimates based upon smaller runs that process 30-days' worth of data. The estimates assume a linear increase in runtime as a function of data size. The increase in runtime for shell scripts for the earlier query is due to an increase in the amount of awk processing logic needed to answer this particular query.

4.1 Visual Analytics Application Overview

To achieve the results we present here, we constructed a custom interactive visual analytics and data mining application and used it to discover a distributed scan. The application makes calls to FastBit [21] to perform network traffic data I/O and queries. FastBit provides the ability to return the number of items that would satisfy a query, and our application leverages this capability to rapidly construct and display histograms. These histograms, combined with visualization and interaction, provide the basis of our visual analytics application. The visualization and rendering portion of our application uses OpenRM Scene Graph [5] and the GUI is built using FLTK [20].

The application's use pattern is as follows. First, the application loads the FastBit "table file" and per-variable indices. The "table file" contains metadata about each of the variables such as data type, min/max values, etc. Next, the user selects any of the variables for display, and the application produces and displays a histogram of counts for each of FastBit's internal bin ranges. This information comes "for free" when using FastBit – no queries are needed to obtain this information. After visual inspection, the application provides the means for forming and posing several different types of queries. The results of a query then appear as a new variable in the application's list of variables and can then be visualized using one of several different dimension-appropriate visualization techniques. A query is formulated by the "cross-product" of range selections that may span an arbitrary number of variables.

The application provides three different mechanisms for specifying such range selections. The first is an interactive selection widget where user specifies a contiguous range of histogram bins using a selection box that is similar to Hochheiser's "timeboxes" [6] but may be applied to any of the twenty-five network traffic variables to produce an $n-$dimensional query. The second method allows a user to specify a starting value, an ending value and a step size. The application will automatically generate histogram bins that are evenly spaced over the user's range selection specified by the typein. The third method is a typein where a user specifies a set of individual histogram bin numbers. This set is simply a list of integer values – bins may be specified in any order, including disjoint, descending, random, etc. The application's query engine combines parameters from each of the three different query specification mechanisms to form an ordered $n-$dimensional query. The field resulting from such a query is an $n-$dimensional dataset of V vector elements. The n spatial axes are formed by the ranges along each of n histograms. The V vector elements are formed by V user-specified histogram bins. Such a formulation of spatial axes and vector components was convenient for this particular case study.

4.2 Network Traffic Analysis

It is necessary to analyze multiple time windows of data to understand the full behavior of a given situation. In this case, the IDS indicated a large number of scanning attempts on tcp port 5554 (Sasser type worm). The larger data set is looked at in order to determine large scale temporal behavior, while the local behavior is looked at from a smaller time window perspective. This sort of activity is typical of an analysis scenario.

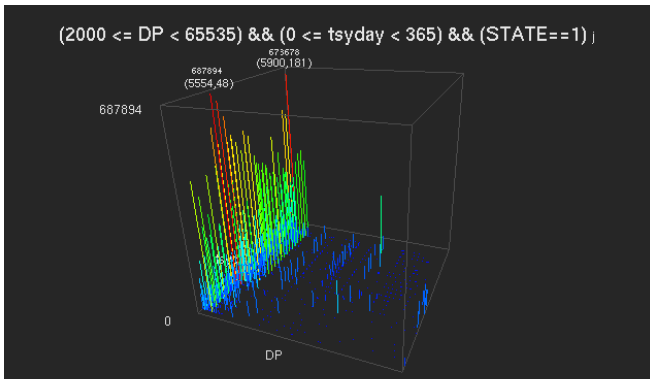

Figure 1: This hisogram shows the number of unsuccessful connection attempts over a 42-week period on (2000 ¡= DP ¡= 65535) with radiation excluded. Based upon the results from this figure, we chose to focus on DP==5554 in Week Three for the rest of the case study. One axis is destination port, the other is time.

First, we will search the 42 weeks' worth of data to identify those ports on which there is a large number of unsuccessful connection attempts. An unsuccessful connection attempt is indicated by a particular value of one of the variables in the traffic logs. Figure 1 shows a plot of counts where one axis is destination port number and the other is time. Each "bin" in the time axis represents one week's worth of activity. The height of each bar represents the number of unsuccessful connection attempts during a particular week on a particular destination port across all addresses within the target destination network. We conclude from this stage of the analysis that there is a high degree of suspicious activity in Week Three on destination port (DP) number 5554.

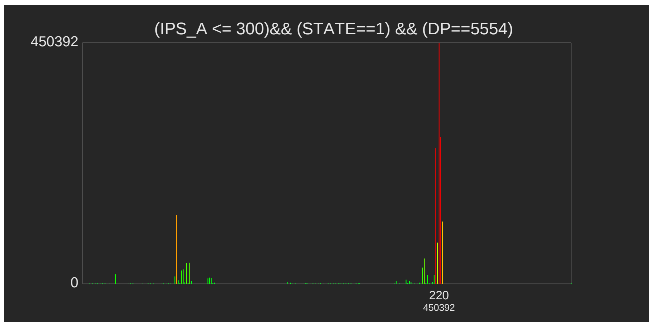

Figure 2: This image shows a 1D plot of IPS_A where DP equals 5554 and STATE is equal to 1. The bin with the largest number of counts is indicated as "220," which means the source of the suspicious activity is from a host or hosts having an IP address where "220" is the first address octet. The bars are colorized such that red bars lie three standard deviations or greater from the mean,which in this case is a value close to zero.

The next step in the analysis process is to determine the addresses of the hosts from which the unsuccessful connection attempts originate. We thus performed the following query over the 42 week dataset: "Find all hosts where $IPS_A <= 300$ && $STATE == 1$ && $DP == 5554$". The horizontal axis in Figure 2's histogram is range of addresses within the Class A octet of the source host IP number. The height of each bar indicates the number of unsuccessful connection attempts from hosts during Week Three on DP 5554 from each Class A address. Note that we are focusing only on the A octet of the source host address at this stage of our analysis. We see a spike for the A octet address of 220, which means that all of the suspicious activity is originating at a host or hosts within a range of IP addresses of 220 in the A octet. To aid the user in quickly and positively identifying anomalies, our application provides a a toggle on each variable's display panel to highlight the the "top N" items in a dataset. In addition, the application provides a statistically-based transfer function so that histogram bars are color-coded by the relationship between a bar's variance and the standard deviation of histogram counts. Bars that are colored red lie about three standard deviations above the mean histogram count.

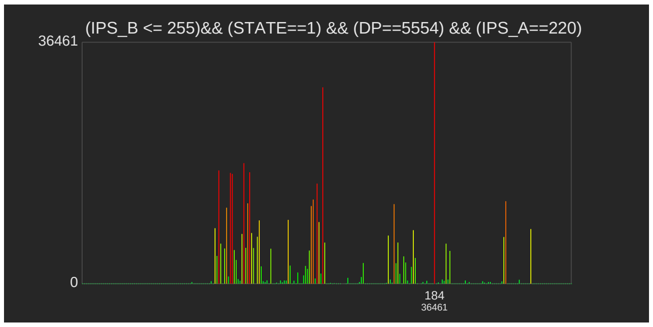

Figure 3: Here we see that from within the 220 Class A octet, the Class B address with the most suspicious traffic is 184. Several potential class B candidates emerge – colored red – and we choose 184 for further investigation. The horizontal axis the the 254 possible Class B addresses, and the vertical axis is the count of unsuccessful connection attempts.

We next refine our search to identify the range within the Class B octet where the suspicious activity originates. To do so, we pose a query to count the number of unsuccessful connection attempts to $DP = 5554$ in Week Three from the 220 Class A address block. The result is the histogram in Figure 3. For the sake of brevity in this case study, we choose to focus on the spike at $IPS_B = 184$ – another interesting candidate appears at about $IPS_B = 128$ that will ignore for the rest of this case study. Using the results from Figures 2 and 3, we know that all unsuccessful connection attempts for this particular investigation originate from within the 220.184 block of IP addresses.

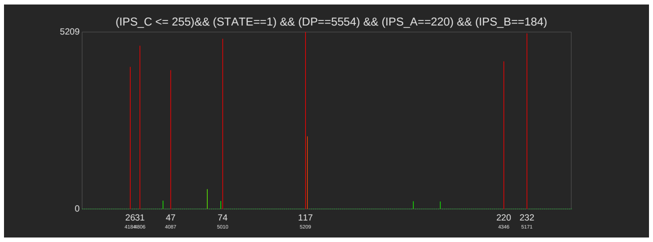

Figure 4: This image shows several spikes in the source Class C address range where STATE==1, DP==5554, $IPS_A == 220$, $IPS_B == 184$ and week=3. We identify the seven largest spikes for further investigation. The horizontal axis is the 254 possible Class C addresses, and the vertical axis the number of unsuccessful connection attempts from the 220.184 network segment.

The next step is to discover the source host address range within the Class C octet. Unlike the source address with Class A and B octets, Figure 4 shows there are unsuccessful connection attempts from hosts in several different addresses in the Class C octet. Seven of these spikes – highlighted by our application – have visually similar levels of suspicious traffic. We hypothesize that hosts on these seven Class C network segments are part of a distributed scan.

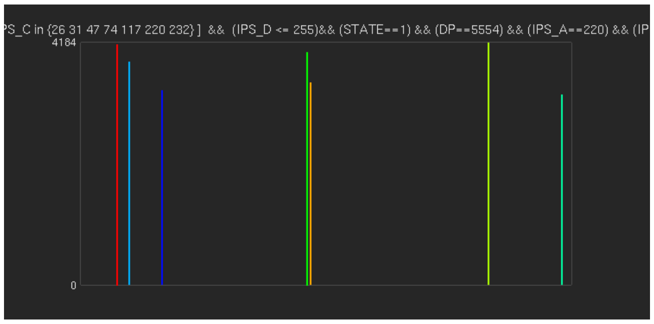

Figure 5: This image shows the class D portion of the addresses of the attacking hosts. We use an iterative compound query to produce this image. The "constant portion" of the query is STATE==1, DP==5554, IPS_A==220, IPS_B==184. The "iterative" portion of the query loops over the seven Class C addresses. The spikes are color-coded by their associated Class C address. The absence of duplicate-colored spikes indicates there are seven unique hosts participating in this particular attack. This color-coding scheme is also used in Figures 6 and 7.

We take this idea one step further to discover the source addresses within the Class D octet. Unlike previous queries, each of which is a single, straightforward range query, we must take a slightly different approach to find the D octet addresses. Our application lets us specify a discrete set of variable values to include as part of a query – in this case, these discrete values are the seven unique Class C addresses seen in Figure 4. The next query is comprised of seven different sub-queries – one for each of the seven unique Class C source addresses. Our aim is to discover all Class D source addresses within each of the seven unique Class C addresses. The result is Figure 5, which indicates traffic from each of the Class D IP addresses corresponds to a single Class C address. In other words, each source host address has a unique Class C and Class D address. At this point, we have positively identified the IP addresses of all the hosts from which the suspicious traffic originates.

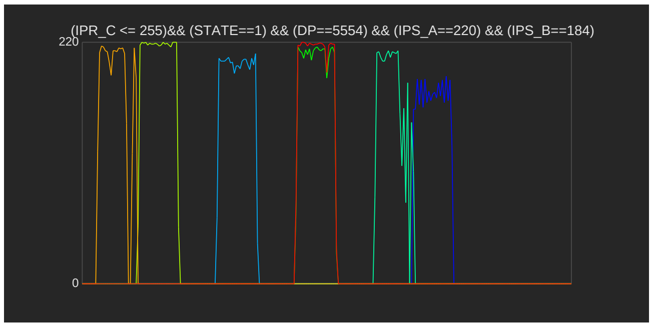

Figure 6: This image shows the set of Class C destination addresses being scanned by each of the source hosts. Note the overlap in one of the destination address ranges. There are seven different colored plots in this image – plots are colored by source host IP number. Two remote hosts are attempting to scan the same block of destination class C addresses as evidenced by the "overlay" of red and green plots. The horizontal axis is the 254 possible destination Class C addresses, and the vertical axis is the number of unsuccessful connection attempts to each of those destination Class C addresses.

The next steps are to look at the problem from a different direction. Rather than focus on identifying the source host addresses – which we have now identified – we are interested in discovering their access patterns through destination IP addresses. Looking first at the destination Class C octet, we see in Figure 6 that each of the seven participating hosts is sending traffic to about 21 or 22 contiguous class C addresses.

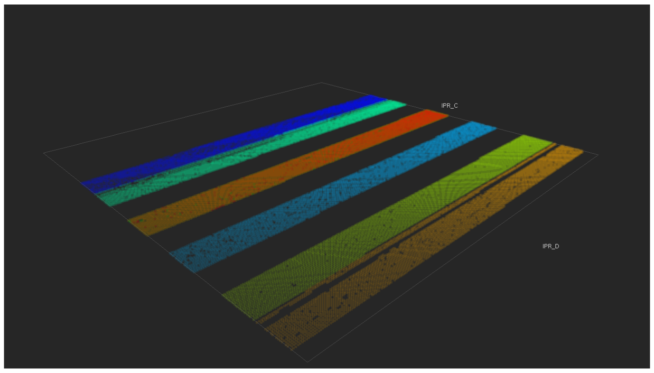

Figure 7: This image shows the coverage in destination IP space by each of the hosts participating in the distributed scan. The axes are the destination C and D address octets. We represent an unsuccessful connection attempt at each (C,D) destination address, and color-code the point based upon the source host address. Two of the remote hosts are scanning the identical block of (C,D) addresses. We use blended transparency and large point primitives to convey this overlap. The overlap appears as the red-green group in the center of the image.

Extending the idea of Figure 6, we we formulate a higher-dimensional query query that produces the histogram shown in Figure 7. Figure 7 uses a 3D scatterplot to confirm the expectation that attacking hosts are attempting to have uniform coverage in the destination address space. As with Figure 6, Figure 7 color-codes each point based upon the IP address of the attacking host. We use transparent point primitives in 3D for this visualization. Looking closely at the "red stripe," it is possible to see the overlay of red points over green points, confirming the behavior we saw previously in Figure 6. Figure 7 shows that each attacking host's scan pattern is (1) to "march through" all Class D addresses for each Class C address, and (2) that each has been assigned a contiguous group of Class C addresses to scan.

4.3 Discussion

Two of the authors on this paper are network security experts whose job duties include operation of production networking facilities. They both contributed to the design and engineering of this new approach, and both are in a good position to evaluate its effectiveness. Their comments, paraphrased below, are particularly insightful.

Because our visual analytics application processes and visualizes statistical information about network traffic data – rather than actual network traffic data – it affords a certain amount of insulation from sensitive information. This approach will allow more people access to network data than otherwise might be possible due to data sensitivity and data size.

The application's general design principles result in a system that is simple to use and easy to understand. The visualizations are very straightforward and require no complex mental mapping to understand. The simple yet effective user interface and interaction design means that this type of interactive analytics very accessible since the learning curve is very shallow and only a passing knowledge of network traffic data is needed to interpret the results.

Both network experts felt this approach was a very useful method for determining the components of a distributed scan. Both were eager to apply the technology to other types of forensic network security projects. Both were excited by the extremely short duty cycle in the data mining process. Neither had ever had the opportunity to explore a single collection of network traffic data of such a large size.

5 Conclusion

In an information-dominated age, the ability to quickly and accurately understand data makes the difference between success or failure in science, business, medicine and education. The work we have presented in this paper takes direct aim at reducing the duty cycle in data-intensive knowledge discovery applications. Our approach – called query-driven visualization and analysis – blends technologies from data management, visualization, analysis and interactive discourse. Our network traffic analysis case study highlights how such a combination provides new capabilities enabling rapid detection and analysis of a distributed network scan.

To reduce the duty cycle in network connection data analysis, we have leveraged several different concepts. First, we accelerate data-mining mechanics by using very efficient index/query scientific data management technology that uses compressed bitmap indices. Second, our visualization techniques are centered about the idea of displaying data distribution information (histograms) that is readily available from the compressed bitmap indexing infrastructure. Focusing visualization in this fashion helps the network analyst to quickly discover "hot spots" in data and to focus subsequent inquiry on those areas. Third, complex multidimensional queries are automatically constructed through "histogram cross-products," which has proven to be a highly effective visual analytics technique for data mining. These concepts were demonstrated on a "hero-sized" dataset consisting of about 2.5 billion records of network connection data collected over a 42-week period.

Acknowledgement

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. The authors wish to acknowledge the contributions by Steven A. Smith, Los Alamos National Laboratory; Brian Tierney and Jason Lee, Lawrence Berkeley National Laboratory to a LBNL Technical Report LBNL-59166, which is a predecessor of the work reported in this paper. The authors wish to particularly thank Smith for a suggestion that evolved into the inspiration for using FastBit's histogram bins as the source of visual analytics in performing guided query formulation.

References

[1] John Bentley. Multidimensional binary search trees used for associative search. *Communications of the ACM*, 18(9):509–516, 1975.

[2] Rene Brun and Fons Rademarkers. Root – an object oriented data analysis framework. In *Proceedings of the AIHENP 1996 Workshop*, pages 81–86, 1997.

[3] Joe Burrescia and William Johnston. Esnet status update. Internet2 International Meeting, 2005.

[4] C.-Y. Chan and Y. E. Ioannidis. Bitmap index design and evaluation. In *Proceedings of the 1998 ACM SIGMOD: International Conference on Management of Data*, pages 355–366, New York, NY, USA, 1998. ACM press.

[5] R3vis Corpoation. OpenRM Scene Graph. http://www.openrm.org, 1999-2006.

[6] Harry Hochheiser and Ben Shneiderman. Visual specification of queries for finding patterns in time-series data. In *Proceedings of Discovery Science*, pages 441–446, 2001.

[7] Van Jacobsen, Craig Leres, and Steven McCanne. tcpdump. ftp://ftp.ee.lbl.gov/, 1989.

[8] Theodore Johnson. Performance measurements of compressed bitmap indices. In *Proceedings of the 25th International Conference on Very Large Data Bases*, September 1999.

[9] Daniel Keim and Hans-Peter Kriegel. Visdb: Database exploration using multidimensional visualization. *IEEE Computer Graphics and Applications*, 14(4):40–49, 1994.

[10] Donald Knuth. *The Art of Computer Programming, 2nd Ed., Volume 3*. Addison-Wesley, 1998.

[11] Anita Komlodi, Penny Rheingans, Utkarsha Ayachit, John Goodall, and Amit Joshi. A user-centered look at glyph-based security visualization. In *Proceedings of the 2005 Workshop on Visualization for Computer Security*, October 2005.

[12] Stephen Lau. The spinning cube of potential doom. *Communications of the ACM*, 47(6):25–26, 2004.

[13] Yarden Livnat, Jim Agutter, Shaun Moon, Robert Erbacher, and Stefano Foresti. A visual paradigm for network intrusion detection. In *Proceedings of the 2005 IEEE Workshop on Information Assurance And Security*, June 17–19 2005.

[14] Steven McCanne, Craig Leres, and Van Jacobsen. libpcap. ftp://ftp.ee.lbl.gov/, 1994.

[15] Jonhathan McPherson, Kwan-Liu Ma, Paul Krystosek, Tony Bartoletti, and Marvin Christensen. Portvis: A tool for port-based detection of security events. In *Proceedings of CCS Workshop on Visualization and Data Mining for Computer Security, ACM Conference on Computer and Communication Security*, October 2004.

[16] Patrick O'Neil. Model 204 architecture and performance. In *Second International Workshop in High Performance Transaction Systems*. Lecture Notes in Computer Science, vol. 359, Springer-Verlag 4059, 1987.

[17] Patrick O'Neil and D Quass. Improved query performance with variant indices. In *Proceedings of ACM SIGMOD International Conference on Management of Data*. ACM Press, May 1997.

[18] Vern Paxson. Bro: A system for detecting network intruders in real-time. In *Proceedings of the 7th USENIX Security Symposium*, January 1998.

[19] Peter Phaal, Sonia Panchen, and Neil McKee. Inmon corporation's sflow: A method for monitoring traffic in switched and routed networks. IETF RFC 3176, http://www.app.sietf.org/rfc/rfc3176.html, 2001.

[20] Easy Software Products. The fast light toolkit. http://www.fltk.org, 2006.

[21] Lawrence Berkeley National Laboratory Scientific Data Management Group. Fastbit. http://sdm.lbl.gov/fastbit, 2005.

[22] Arie Shoshani, Luis Bernardo, Henrik Nordberg, Doron Rotem, and Alex Sim. Multidimensional indexing and query coordination for tertiary storage management. In *Proceedings of the 11th International Conference on Scientific and Statistical Database Management*. IEEE Computer Society 214225, July 1999.

[23] Kurt Stockinger, Dirk Duellmann, Wolfgang Hoschek, and Erich Schikuta. Improving the performance of high-energy physics analysis through bitmap indices. In *Proceedings of the 11th International Conference on Database and Expert Systems Applications*, 2000.

[24] Kurt Stockinger, John Shalf, Kesheng Wu, and E. Wes Bethel. Query-driven visaulization of large data sets. In *Proceedings of IEEE Visualization*, pages 167–174, October 2005.

[25] Kurt Stockinger, Kesheng Wu, Rene Brun, and P. Canal. Bitmap indices for fast end-user physics analysis in root. In *Nuclear Instruments and Methods in Physics Research, Section A – Accelerators, Spectrometers, Detectors and Associated Equipment*. Elsevier, to appear.

[26] Cisco Systems. Cisco netflow collection engine. http://www.cisco.com/en/US/products/sw/netmgtsw/ps1964/, 2005.

[27] James J. Thomas and Kristin A. Cook eds. *Illuminating the Path – The Research and Development Agenda for Visual Analytics*. IEEE Computer Society Press, 2005.

[28] Benjamin Uphoff and P. Criscoulo. A framework for collection and management of intrusion detection data sets. In *Proceedings of the 16th Annual FIRST Conference on Computer Security Incident Handling*, June 2004.

[29] Kesheng Wu, Ekow Otoo, and Arie Shoshani. A performance comparison of bitmap indices. In *Proceedings of the ACM CIKM International Conference on Information and Knowledge Management*. ACM 559561, November 2001.

[30] Kesheng Wu, Ekow Otoo, and Arie Shoshani. On the performance of bitmap indices for high cardinality attributes. In *In Proceedings of the International Conference of Very Large Data Bases*, pages 24–35, 2004.